PROBLEM 1:  (What is the output? (18 pts))

For the following code, write the output to the right of each print statement. The output for the first three print statements is already shown.

```python
seta = set([6, 3, 2])
print(seta)  # {6, 3, 2}
print(list(seta))  # [6, 3, 2]
print(sorted(seta))  # [2, 3, 6]

#-----------------------------
seta = set([4, 2, 7, 3, 4, 3, 3, 7, 5])
seta.add(7)
seta.add(8)
print(sorted(seta))

#-----------------------------
seta = set([5, 7, 6, 5, 5])
seta.remove(5)
print(sorted(seta))

#-----------------------------
seta = set([8, 5, 3, 4])
setb = set([6, 3, 8, 5])
print(sorted(seta & setb))
print(sorted(setb ^ seta))
print(sorted(setb - seta))

#-----------------------------
lst = [8, 9, 4, 3]
dict = {"A": 6, "W": 3, "S": 4}
print(sorted(dict.keys()))
print([dict[k] for k in dict if dict[k] in lst])
dict["P"] = 3
dict["S"] = 6
print(sorted(dict.keys()))
print(sorted(dict.values()))
```

OUTPUT:

```
{6, 3, 2}
[6, 3, 2]
[2, 3, 6]

-----------------------------

[4, 2, 7, 3, 4, 3, 3, 7, 5]

-----------------------------

[5, 7, 6, 5, 5]

-----------------------------

[8, 5, 3, 4]

-----------------------------

[6, 3, 8, 5]

-----------------------------

[8, 9, 4, 3]
```
For problem Part A, give the output. For each of the remaining problems, assign the variable `result` to a Python expression that calculates the answer. That means if we changed the list given, your code would still calculate the correct answer.

Each of these must be written in one line and include a list comprehension.

Here is an example.

The variable `result` should calculate the list of words from list `vehicles` that have the letter 'a' in their word. Assume each string in the list is one word that is lowercase.

Using the list `vehicles` below, `result` would calculate the list:

```
['train', 'airplane', 'car', 'longboard']
```

```python
vehicles = ['train', 'airplane', 'car', 'truck', 'bike', 'longboard']
result = 
```

ANSWER:
```
result = [w for w in vehicles if 'a' in w]
```

PART A (2 pts)

In this part we have given you the list comprehension and you need to give the output when this code executes.

```python
lst = [4, 1, 6, 5]
result = [2*n+3 for n in lst]
print(result)
The output is:
```

3
PART B (4 pts)
The variable result should calculate the list of words that start with the letter 'c' from the list lst. You can assume all words in lst contain at least one letter, and the words in result are in the same order their corresponding words are in lst.
Using the list lst below, result would calculate the list: ['car', 'cat', 'cheese']

lst = ['train', 'car', 'a', 'dog', 'cat', 'cheese', 'apple']

result =

PART C (4 pts)
The variable result should calculate the list of numbers from the list lst that are less than 10 or greater than 50. The numbers in result should appear in the same order they appeared in lst.
Using the list lst below, result would calculate the list: [8, 3, 78, 97, 6]

lst = [21, 8, 45, 3, 31, 78, 97, 6, 49]

result =
PART D (4 pts)

The variable `result` should calculate the list of abbreviations for each word in the list `lst`, where an abbreviation is the first two letters of the word followed by the last letter of the word. If a word contains only one letter, than its abbreviation is the first letter and the last letter. You can assume all words in `lst` contain at least one letter, and the abbreviations in `result` are in the same order their corresponding words are in `lst`.

Using the list `lst` below, `result` would calculate the list: `['bat', 'aa', 'inn', 'red', 'moe', 'trn', 'att']`

`lst = ['bat', 'a', 'in', 'red', 'mouse', 'train', 'at']`

`result =`
For each of these problems, calculate the answer with code, such that the variable `result`'s value should be the resulting answer.
If we changed the list(s) or dictionaries given, your code should still calculate the correct answer.

You can write your answer in more than one line, and with more than one variable, but be sure result’s value should be the answer.

Here is an example.
Calculate the list of words from list `vehicles` that have the letter ’a’ in their word. Assume each string in the list is one word that is lowercase. The answer should be stored in the variable `result`.
Using the list `vehicles` below, `result`’s value would be

`['train', 'airplane', 'car']`

```python
vehicles = ['train', 'airplane', 'car', 'truck', 'bike']
result =
```

**ANSWER:**

```python
result = [w for w in vehicles if 'a' in w]
```

**ALTERNATIVE ANSWER:**

```python
result = []
for w in vehicles:
    if 'a' in w:
        result.append(w)
```

With both of these the answer is calculated and stored in result.

This problem has four parts. The four problems start on the next page.
PART A (4 pts)
Given the list of strings named `words`, where each string is one word, calculate a list of tuples, where each tuple has two items, 1) the first letter of the word and 2) the word. The tuples should appear with the words in the same order they are in the list `words`. The answer should be stored in the variable `result`.

Using the list `words` below, `result`'s value would be: `[('b', 'board'), ('l', 'letter'), ('r', 'run'), ('b', 'ball')]`

Write your code below and be sure result’s value is the answer.

```python
words = ['board', 'letter', 'run', 'ball']
```

PART B (4 pts)
You are given the list of strings `lsta`, and the list of strings `lstb`, where each string in both lists is one word. Calculate a sorted list of the unique words from `lsta` that are not in `lstb`. The answer should be stored in the variable `result`.

Using the lists below, `result` would be: `['bush', 'rake', 'tree']`.

Write your code below and be sure result’s value is the answer.

```python
lsta = ['leaf', 'tree', 'bush', 'leaf', 'tree', 'rake']
lstb = ['wind', 'leaf', 'beetle', 'wind', 'leaf']
```
PART C (4 pts)
You are given the list of strings named snacks that are snacks and the list of integers named numbers. Assume the lists are the same length. The kth string in the list numbers is the number of orders for the kth item in the list snacks.

Calculate a list of tuples, where each tuple has two items, 1) a word from the list snacks and 2) one more than the specified order for that snack. The tuples should appear with the snacks in the same order they are in the list snacks. The answer should be stored in the variable result.

Using the list lst below, result’s value would be: [('pretzels', 6), ('cookies', 5), ('chips', 3), ('carrots', 6)]

Note that the numbers in the list of tuples reflect one more than the original order for each snack.

Write your code below and be sure result’s value is the answer.

snacks = ['pretzels', 'cookies', 'chips', 'carrots']
numbers = [5, 4, 2, 5]
PART D (4 pts)

You are given the list of strings `words` where each string in the list is one word, and there is at least one word in the list. Calculate a list of words from the list `words` such that you include a word only if the word immediately before it in the list comes before it in alphabetical order. The first word in `words` is always included in the new list as it does not have a word that comes before it. The answer should be stored in the variable `result`, and the words in `result` should be in the same order their corresponding words are in `words`.

Using the lists below, `result` would be: `['letter', 'car', 'run']`.

Note that 'letter' is included because it is the first word in the list `words`, 'car' is included because the word immediately before it, 'board', comes before it in alphabetical order, and 'run' is included because 'car' comes before it in alphabetical order.

Write your code below and be sure `result`'s value is the answer.

```python
words = ['letter', 'board', 'car', 'run', 'ball']
```
PROBLEM 4 : (What is selling at the market? (24 pts))

This problem is about data related to items sold at a farmers market.
There are four functions to write in this part. Your functions should work for any valid data, not just the examples shown.
The first three problems have datalist as one of the parameters. The parameter datalist is a list of lists, with each inner list having the following three items: 1) a string representing the name of a seller, 2) an integer representing the age of the seller, and 3) a unique list of strings of items the seller sells at the market. We will assume seller’s names are unique.
For example, assume datalist is the lists of lists shown below. Note that the first item in the first inner list has the seller Claire who is 50 years old. She sells eggs, cheese and chicken at the market. The second inner list has the seller Fred who is 67 years old. He sells carrots, potatoes and turnips at the market.

datalist = [['Claire', 50, ['eggs', 'cheese', 'chicken']],
            ['Fred', 67, ['carrots', 'potatoes', 'turnips']],
            ['Myra Sue', 72, ['broccoli', 'carrots', 'lettuce', 'kale']],
            ['Jingxing', 48, ['lettuce', 'kale', 'beets', 'spinach', 'tomatoes', 'carrots']],
            ['Sauren', 55, ['tomatoes', 'plums', 'apples', 'peaches']],
            ['Navi', 69, ['beef jerky', 'chicken', 'apples', 'beets']]]

Go to the next page to start Part A of this problem.
Part A (6 pts)
Write the function named `overAge` that has two parameters. The first parameter is named `datalist`, which is a list of lists in the format described earlier, and the second parameter is an integer named `age`.

We repeat the format of `datalist`. The parameter `datalist` is a list of lists, with each inner list having the following three items: 1) a string representing the name of a seller, 2) an integer representing the age of the seller, and 3) a unique list of strings of items the seller sells at the market.

This function returns a sorted list of the unique names of sellers from `datalist` who are over the age of the parameter `age`.

For example, assume `datalist` is the list of lists shown on the first page of this problem. We give several examples of calls to this function.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>overAge(datalist, 68)</code></td>
<td>['Myra Sue', 'Navi']</td>
</tr>
<tr>
<td><code>overAge(datalist, 50)</code></td>
<td>['Fred', 'Myra Sue', 'Navi', 'Sauren']</td>
</tr>
</tbody>
</table>

Complete the function below.

```python
def overAge(datalist, age):
```
Part B (6 pts)
Write the function named `whoSells` that has three parameters. The first parameter is named `datalist`, which is a list of lists in the format described earlier, the second parameter is a string named `item1`, and the third parameter is a string named `item2`.

We repeat the format of `datalist`. The parameter `datalist` is a list of lists, with each inner list having the following three items: 1) a string representing the name of a seller, 2) an integer representing the age of the seller, and 3) a unique list of strings of items the seller sells at the market.

This function returns a sorted list of sellers who sell either `item1` or `item2` or both at the market. We give two examples of calls to this function.

The call `whoSells(datalist, 'carrots', 'chicken')` returns the list `['Claire', 'Fred', 'Jingxing', 'Myra Sue', 'Navi']`.

The call `whoSells(datalist, 'beets', 'spinach')` returns the list `['Jingxing', 'Navi']`

Complete the function below.

```python
def whoSells(datalist, item1, item2):
```
Part C (6 pts)

Write the function named `itemsSold` that has two parameters, The first parameter is named `datalist`, which is a list of lists in the format described earlier, and the second parameter is an integer named `length`.

We repeat the format of `datalist`. The parameter `datalist` is a list of lists, with each inner list having the following three items: 1) a string representing the name of a seller, 2) an integer representing the age of the seller, and 3) a unique list of strings of items the seller sells at the market.

This function returns a sorted set of the unique names of items that are sold at the market whose item name is of length less than the parameter `length`. We give two examples of calls to this function.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>itemsSold(datalist, 6)</code></td>
<td><code>['beets', 'eggs', 'kale', 'plums']</code></td>
</tr>
<tr>
<td><code>itemsSold(datalist, 7)</code></td>
<td><code>['apples', 'beets', 'cheese', 'eggs', 'kale', 'plums']</code></td>
</tr>
</tbody>
</table>

Complete the function below.

```python
def itemsSold(datalist, length):
```
Part D (6 pts)
Write the function named `processFile` that has one parameter named `filename`, that is the name of a file with data described in the following format. This function reads in the file and returns a list of lists described on the first page of this problem.

Each line in `filename` has the following format. 1) The name of a seller, 2) a colon, 3) an integer representing an age, 4) a colon, 5) the name of items the person sells, with any consecutive items separated by a dash. Here is an example of a file.

Claire:50:eggs-cheese-chicken
Fred:67:carrots-potatoes-turnips
Myra Sue:72:broccoli-carrots-lettuce-kale
Jingxing:48:lettuce-kale-beets-spinach-tomatoes-carrots
Sauren:55:tomatoes-plums-apples-peaches
Navi:69:beef jerky-chicken-apples-beets

In this file, the first line represents the seller Claire, who is 50 years old. She sells eggs, cheese and chicken at the market. The last line of the file is the seller Navi who is 69 years old. Navi sells four items at the market, beef jerky, chicken, apples and beets.

The function `processFile` opens and reads the file and returns the list of lists described earlier. For example if the file above is passed to `processFile`, then `processFile` returns the list below.

```
[ ['Claire', 50, ['eggs', 'cheese', 'chicken']], # only first two lists shown
  ['Fred', 67, ['carrots', 'potatoes', 'turnips']], ...]
```

Complete the function below that has been started for you.

```python
def processFile(filename):
    f = open(filename)
    datalist = [ ]
    for line in f:
        # Your code here
```

15